SMONE: A Session-based Recommendation Model Based on Neighbor Sessions with Similar Probabilistic Intentions

BOHAN JIA, JIAN CAO, and SHIYOU QIAN, Department of Computer Science and Engineering, Shanghai Jiaotong University, China
NENGJUN ZHU, School of Computer Engineering and Science, Shanghai University, China
XIN DONG, LIANG ZHANG, LEI CHENG, and LINJIAN MO, Ant Group, China

A **session-based recommendation system (SRS)** tries to predict the next possible choice of anonymous users. In recent years, **graph neural network (GNN)** models have been successfully applied to SRSs and have achieved great success. Using GNN models in SRSs, each session graph is processed successively to obtain the embedding of the node (i.e, each action on an item), which is then imported into the prediction module to generate recommendation results. However, solely depending on the session graph to obtain the node embeddings is not sufficient because each session only involves a few items. Therefore, neighbor sessions have been used to extend the session graph to learn more informative node representations. In this paper, we introduce a *S*ession-based recommendation *MO*del based on *N*eighbor sessions with similar probabilistic int *E*ntions(SMONE). SMONE models the intentions behind sessions in a probabilistic way and retrieves the neighbor sessions with similar intentions. After the neighbor sessions are found, the target session and its neighbor sessions are modeled as a hypyergraph to learn the contextualized embeddings, which are combined with item embeddings through GNN to produce the final item recommendations. Experiments on real-world datasets prove the effectiveness and superiority of SMONE.

CCS Concepts: • **Information systems → Recommender systems;**

Additional Key Words and Phrases: Session-based recommender systems, neighbor sessions, probabilistic intentions

ACM Reference format:
Bohan Jia, Jian Cao, Shiyou Qian, Nengjun Zhu, Xin Dong, Liang Zhang, Lei Cheng, and LinJian Mo. 2023. SMONE: A Session-based Recommendation Model Based on Neighbor Sessions with Similar Probabilistic Intentions. *ACM Trans. Knowl. Discov. Data.* 17, 8, Article 111 (May 2023), 22 pages.
https://doi.org/10.1145/3587099

This work is supported by China National Science Foundation (Grant Nos. 62202282 and 62072301). This work is also partially supported by the Program of Technology Innovation of the Science and Technology Commission of Shanghai Municipality Granted No. 21511104700).
Authors' addresses: B. Jia, J. Cao (corresponding author), and S. Qian, Department of Computer Science and Engineering, Shanghai Jiaotong University, 800 Dongchuan Road, Shanghai, China, 200240; emails: bohan_ieee@163.com, caojian@sjtu.edu.cn, qshiyou@cs.sjtu.edu.cn; N. Zhu, School of Computer Engineering and Science, Shanghai University, 333 Nanchen Road, Shanghai, China, 200444; email: zhu_nj@shu.edu.cn; X. Dong, L. Zhang, L. Cheng, and L. Mo, Ant Group, 569 Xixi Road, Hangzhou, Zhejiang, China, 310013; emails: zhaoxin.dx@antgroup.com, zhuyue.zl@antgroup.com, lei.chenglei@antgroup.com, linyi01@antgroup.com.

1556-4681/2023/05-ART111 $15.00
https://doi.org/10.1145/3587099

1 INTRODUCTION

In recent years, an increasing number of enterprises are utilizing recommendation systems to increase user interaction and enrich shopping potential [4, 34]. Some recommender systems leverage the side information of users or items [25, 35]. Unfortunately, this side information is not always available. As an important paradigm of recommender systems, **session-based recommender systems (SRSs)** have become popular in both academic research and business applications. In SRSs, only anonymous users' actions (actions and items are used interchangeably in this article) in one session are obtained and their next actions are predicted in a timely manner. Because a user's long-term preferences are not available, since they are anonymous, SRSs try to learn short-term but dynamic user preferences based on their session contexts.

Because of its ability to model complex dependencies, **graph neural networks (GNNs)** have become the main approaches and have achieved great success, because session data can easily be represented as a graph [43]. In the GNN-based approach, each session graph is processed successively to obtain the embedding of the node (i.e, each action on an item), which is imported into the prediction module of SRSs. However, solely depending on the session graph to obtain the node embedding is not sufficient. Some researchers focused on the neighbor sessions. Neighbor sessions can provide more information but, at the same time, they also bring two challenges to the SRSs. The first challenge is how to find appropriate neighbor sessions. The second challenge is how to learn from the neighbor sessions.

Users' behaviors are driven by intentions. Therefore, sessions with similar intentions can reveal related action patterns and also have a greater chance of including similar items. Some studies have tried to extract intentions from sessions and make use of them in a different way [8, 9, 20, 28, 51, 54, 55]. In particular, in Reference [28] intentions are also applied to retrieve neighbor sessions. In these studies, intentions are represented as the category, keywords, or embeddings of items. However, the concept of intention is complex, and these explicit representations oversimplify the meaning of intentions. In general, when a user browses products, his intention is a mixture of some intended actions. For example, in Figure 1, there are several sessions. The intentions behind these sessions are represented by a probability vector. Although these vectors are different, they are similar to the corresponding vector of the target session. Therefore, we can take them as the neighbor sessions of the target session. Probabilistic intention modeling for sessions has a high tolerance to the signals brought by miscellaneous items. For example, in Figure 1, the first session includes a pen, which can be regarded as a noise signal in terms of the other items in this session. However, the intentions behind these sessions can still be revealed correctly. Therefore, we propose a ***S*ession-based recommendation *MO*del based on *N*eighbor sessions with Similar Probabilistic Int*E*ntions (SMONE)**.

After the neighbor sessions are identified, a graph can be constructed for each session and is processed successively to learn the embeddings of items, similar to DGTN [53]. In Reference [28], the weighted sum of each neighbor session's representation is integrated with the target session. All these approaches treat each neighbor session and target session independently and neglect the potential connections among them. In SMONE, we try to incorporate the neighbor sessions and the target session into a unified hypergraph and learn contextualized item embedding directly.

The main contributions of SMONE can be summarized as follows:

- We propose a method to model the probabilistic intentions behind sessions and obtain the neighbor sessions in terms of the intention similarity.
- We use a unified hypergraph to model the potential connections between the neighbor sessions and the target session to derive the contextualized item embeddings.
- We evaluate our model on three real-world datasets. Extensive experiments show the effectiveness and superiority of SMONE.

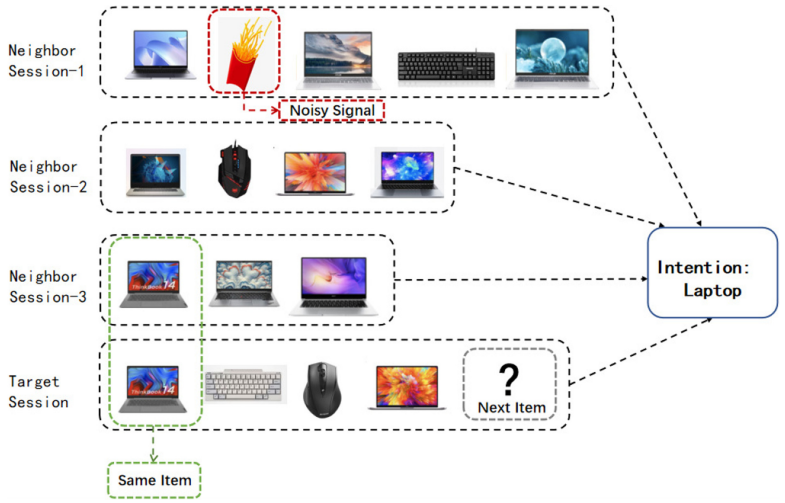


Fig. 1. The sessions with the probabilistic intentions.

The rest of this article is structured as follows: In Section 2, we introduce the background and related work. In Section 3, we describe the framework of SMONE. The approach to identify neighbor sessions based on probabilistic intention modeling is introduced in Section 4. Section 5 describes the other key components of SMONE in detail. The experiments and result analysis are presented in Section 6. The article is concluded in Section 7.

2 BACKGROUND AND RELATED WORK

The problem of session-based recommendation aims to predict user actions based on anonymous sessions. The biggest challenge in SRSs is that it cannot connect multiple sessions in terms of their user identities directly. According to the techniques applied, the approaches for SRSs can be divided into conventional approaches, latent representation approaches, and deep neural network approaches [39]. In addition, graph neural networks, a special deep neural network model, are becoming the major models for SRSs.

2.1 Conventional SRSs

Conventional approaches adopt conventional data mining or machine learning techniques to learn the dependencies embedded in session data. For example, **Markov chains (MC)** can be utilized to model the sequence of actions and predict a user's next actions based on the session context [45]. MC-based methods focus on modeling transitions between two consecutive items and cannot represent the complex dependencies among items in a session.

2.2 Latent Representation-based SRSs

Latent representation approaches try to learn a low-dimensional latent representation for each action within sessions with a matrix factorization model [32] or shallow neural network model [40]. Then the informative representations that encode the dependencies between actions can be used in recommendation. Factorization models suffer from the data sparsity problem and cannot capture higher-order and long-term dependencies, while it is difficult for shallow neural network models to model ordered or heterogeneous sessions.

2.3 Deep-learning-based SRSs

In recent years, neural network models have played an increasingly important role in SRSs. Specifically, **recurrent neural networks (RNNs)** have been utilized in many studies, as they can process sequential data [14, 23]. In addition, **multi-layer perceptron (MLP)** networks [24] and **convolutional neural networks (CNNs)** [1] have also been applied in SRSs. Since sessions can be represented as a sequence of items, RNNs have significantly improved the performance of SRSs [14]. There are many studies that apply RNNs to SRSs. For example, in Reference [14], novel ranking loss functions are designed for RNNs that are applied to SRSs. Moreover, the structures of ordinary RNN models are extended to adapt to the requirements of SRSs, such as the variational recurrent model [41]. Attentions are also introduced to RNN models for SRSs [21]. However, RNN models only focus on the consecutive time patterns of the transitions of item choices and cannot capture more complicated item transition patterns. Therefore, network models with different structures are designed to encode more information. FWSBR [17] is proposed to integrate item attributes and user–item interactions to make sustainable session-based recommendation. SASRec [19] stacks multiple blocks of self-attention layers and point-wise feed-forward layers, which can assign weights to previous items and sum them up. L2 normalization and dropout are adopted in NISER [13] to deal with the long-tail problem and overfitting problem, respectively. In addition, feature encoder and a self-attention layer are introduced to aggregate item features together as the session feature. SSRM [12] considers a specific user's historical sessions and applies the attention mechanism to combine them. SPLIT [33] is a sequential recommender model via decomposing and modeling user independent preferences in a sequence. These models only consider the session as a totally random set.

2.4 GNN-based SRSs

Recently, GNNs have become the mainstream approach of SRSs in the existing research. Using GNNs, sessions can be modeled as graphs and item embeddings are learned with GNNs. Then, the item embeddings and session representations based on item embeddings are inputted into the prediction modules of SRSs [43]. Many different GNN models have been proposed to improve the performance of SRSs. For example, Qiu et al. [30] proposed a model named the **Full Graph Neural Network (FGNN)** to learn the inherent order of the item transition pattern and compute a session-level representation to generate recommendations. GARG [44] applies a graph convolutional neural network and the attention mechanism to provide users with appropriate new points of interest. LESSR [7] proposes an edge-order preserving aggregation scheme based on GRU and a shortcut graph attention layer to address the lossy session encoding problem and effectively capture long-range dependencies, respectively. CGSNet [36] introduces contrastive learning to overcome data sparsity in session-based recommendation.

Although these GNN models can capture the item dependencies in a session effectively, the information obtained from the current session is inadequate. First, the length of a session is often not long enough to provide adequate information to learn the users' interests. Second, user actions in sessions may contain noisy signals, i.e., a user may click an item randomly. The noisy signals may lead to more errors in shorter sessions.

To obtain richer information, an extended graph can be built on historical sessions and the current session for item embedding learning. Quadrana et al. [31] proposed HRNN, which applies a recurrent architecture to aggregate the average feature of all sessions from the user's history. Bai et al. [2] proposed ANAM, which uses an attention model to combine different sessions. In Reference [47], Xu et al. presented the graph context self-attention model GC-SAN, which uses the graph neural network and the self-attention mechanism to learn the long-term dependence between items in the session sequence. Yu et al. [50] proposed TAGNN to adaptively activate

user interests by considering the relevance of historical behaviors given a target item. LDGC-SR [29] integrates Long-range dependencies and global context information for session-based recommendation. In the GCN-GNN model, in addition to the session graph, a global graph is used to learn the global-level item embedding by modeling pairwise item-transitions over all sessions [42]. Reference [38] extracts both intra- and inter-session dependencies not only from the session information but also the side information. However, the size of the global graph is too large, and noisy signals are introduced into the learning process. CGL [27] employs the Gated Graph Neural Networks to learn item embeddings. The model is trained by both the main supervision as well as the self-supervision signals. The sequential order provides supervisions, and the global graph sessions provide self-supervisions.

Some works attempt to make use of information from other relevant sessions to construct the graph, which is called *neighbor sessions* [22, 53]. Neighbor sessions can provide more information and at the same time, they also bring two challenges to the SRSs. The first challenge is how to find appropriate neighbor sessions. The second challenge is how to learn from the neighbor sessions.

The easiest way to retrieve neighbor sessions is to find historical sessions, a strategy that is implemented in DGTN [53]. After finding the neighbor sessions, DGTN integrates the target session and its neighbor sessions into a single graph. However, in practice, it is difficult to find enough neighbor sessions because of the sparseness of item interactions in SRSs. In Reference [22], the neighbor sessions are defined as sessions that have similar item tags in the target session. However, tag information is not always available. Moreover, the relevancy of neighbor sessions cannot be assured when only a small set of tags are defined in the system, because a tag will be shared by quite a few items with different properties. Some researchers have explored the concept of intention in session-based recommendation. Cui et al. [9] propose a model that considers category information as user intention. In Reference [8], the intention is also represented by the category information. In Reference [20], keywords are used as intentions. Another work [10] regards the last clicked item and category information as intentions. Zhang et al. [51] combine dynamic intentions and static intentions by constructing the current session as a digraph and undigraph. They use the last node's vector of the digraph to represent the user's dynamic intentions. The user's static intentions are obtained by an attention mechanism above the node vectors of the undigraph. Although using these explicit concepts as intentions is explainable, it neglects the latent property of intentions. In contrast, in our model, intentions are directly modeled as probability factors.

Reference [28] retrieves the neighbor sessions in terms of the intention similarity and combines the representations of the current session and the neighbor sessions to produce the final item recommendations. CSRM [37] applies memory networks to contain auxiliary information from neighbor sessions. The ideas underpinning our model and theirs are similar. However, they use the last predicted item as the intention and we use probability factors to represent the intentions. Moreover, in Reference [28], weighted neighbor session representations and the target session representation are integrated directly through a gated fusion layer. In our model, we apply a unified hypergraph to model all the neighbor sessions and target sessions and to learn more informative representations.

3 SMONE: A SESSION-BASED RECOMMENDATION MODEL BASED ON NEIGHBOR SESSIONS WITH SIMILAR PROBABILISTIC INTENTIONS

In this section, we formalize the problem first and then introduce the pipeline of SMONE.

3.1 Problem Statement

The item set $V = \{v_1, v_2, \ldots, v_N\}$ represents all of the unique items. An anonymous session s_d can be denoted as a list $s_d = \{v_{d,1}, v_{d,2}, \ldots, v_{d,n_d}\}$ ordered by timestamps, where $v_{s,i} \in V$ is an item

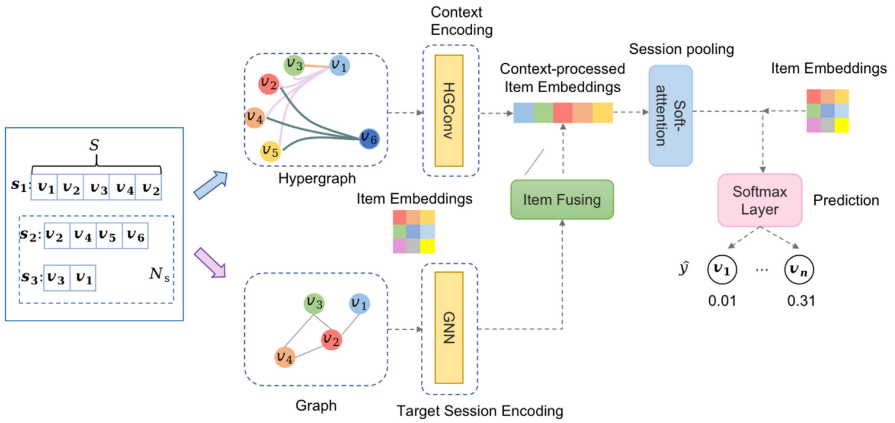


Fig. 2. The pipeline of SMONE.

that is clicked by the user in this session. There is a historical session set S_h and a target session $s_o = \{v_{o,1}, v_{o,2}, \ldots, v_{o,n_o}\}$ and the goal of SRSs is to predict the next item v_{o,n_o+1} in which the user may be interested.

3.2 Overview of the Pipeline of SMONE

SMONE consists of four steps, as shown in Figure 2. The first step is to find neighbor sessions for the target session. Specifically, this step utilizes the topic model to mine latent intentions of sessions and clusters sessions with similar intentions. The second step is to learn item embeddings through two channels: a target session channel and a context channel. The target session channel models the target session as a graph to learn the embeddings of the items. The context channel learns the context information through a hypergraph based on the target session and the neighbor sessions. In the third step, the node embeddings of the current session are concatenated with contextualized node embeddings. The concatenated embeddings and the current item embeddings are fused by a linear layer. Then the fused embeddings are processed to obtain the session representation through a soft-attention module. Finally, the probability $\hat{Y}$ of all candidate items is obtained by multiplying each candidate by the target session representation. $y_i \in \hat{Y}$ is the output of the model, which is the relevance score of the corresponding item. We can select candidate items with the top-k values for the recommendation.

The first key technique of SMONE is to find the informative neighbor sessions using a specifically designed topic model, which captures item dependency relationships in a session. The second key technique is to learn the target session representation in a more comprehensive way. We introduce these two techniques in the next two sections in detail.

4 NEIGHBOR SESSIONS WITH SIMILAR PROBABILISTIC INTENTIONS

A session comprises a sequence of items, which can be regarded as a piece of text. Topic models have been widely used to uncover hidden topics from the text corpus [3]. In topic models, documents are modeled as mixtures of topics and each topic is a probability distribution over words. After applying a topic analysis model such as PLSA [16] and LDA [5], the topic components and mixture coefficients of topics for each document can be learned. By treating each session as a text and topics as intentions directly, sessions can be modeled as mixtures of intentions and each intention is a probability distribution over items. Unfortunately, the lengths of sessions are

Table 1. Notation Table

Notation	Description
α	hyper parameters for intention Dirichlet distribution
β	set of hyper parameters for item Dirichlet distribution
Θ	represents the probability of sessions generated by all intentions
Φ	represents the intention-item distribution
n_d	represents the length of session s_d
$\phi_{z_d, v_{d,j-1}}$	polynomial distribution
$\delta_{a,b}$	denotes the Kronecker delta function
$\phi_{k,r}$	intention-specific item distribution
$\beta^k_{(.),r}$	vector $\beta^k_{(.),r} = \{\beta_{k,r.l}\}_{l=1}^N$
$\beta_{k,r}$	hyper parameters for item Dirichlet distribution
$\eta^k_{(.),r}$	vector $\eta^k_{(.),r} = \{n_{k,r.l}\}_{l=1}^N$
$n_{k,r}$	the times of intention k associated with the item r in the whole data
$n_{k,r.l}$	the times of intention k associated with the item pair $r.l$ in the whole data
Δ	normalization factor $\Delta = \frac{\Gamma(\sum_{i=1}^I x_i)}{\prod_{i=1}^I \Gamma(x_i)}$
τ_k	the number of sessions owned by intention k
$\mathbf{T}$	the set of number of sessions owned by intentions
$\mathbf{A}$	the set of Dirichlet priors owned by intentions
Z_{-d}	Z-$\{z_d\}$
S	the set of all sessions

generally very short and the item co-occurrence patterns in each session are very sparse, so traditional models including PLSA and LDA cannot work well.

The good news is that some topic models have been proposed for short texts in recent years. In particular, the researchers utilize a mixture of unigrams [26, 52] for short text classification. Moreover, in Reference [48], a **biterm topic model (BTM)** is proposed for short texts where a biterm represents a co-occurring word-pair. In some of the experiments that have been conducted, BTM has proved its effectiveness in working on short text. However, these models follow the assumption of bag-of-words but ignore the dependency information, i.e., they conduct statistics on words to form a co-occurrence matrix without considering the dependency relationship. In BTM, the two words that compose the word-pair in the corpus may be separated by several words and they may have no direct relationships. In session-based recommendation, there are casual and time-dependence relations behind the items in a session, because after users browse an item they often move to the item that is related to the previous one. Therefore, our probabilistic intention model for sessions restricts the item-pair to the one whose items appear sequentially in a session.

For convenience of description, we use the notations in Table 1 to describe the probabilistic intention model.

4.1 Probabilistic Intention Modeling for Sessions

For session $s_d = \{v_{d,1}, v_{d,2} \ldots, v_{d,n_{d-1}}, v_{d,n_d}\}$, we assume that all the items appear in session s_d because of some inherent factors. We call this factor as intention and the selection of all the items in s_d is mainly driven by the intention.

Suppose there are a total of K primitive intentions. The intention distribution for session s_d is a K-dimensional probability vector $\mathbf{P}_{s_d}$. To obtain the intention distribution for each session, first, we suppose each session is associated with an intention and the corresponding intentions of all sessions are represented by $Z = \{z_1, \ldots, z_M\}$, where M represents the total number of sessions in S.

Suppose α and β are the Dirichlet priors. Θ represents the probability of sessions generated by all intentions and Φ represents the intention-item distribution, respectively. For the whole session

set S, the likelihoods of Z, Θ, and Φ can be expressed as follows:

$$\begin{aligned} p(S, Z, \Phi, \Theta|\alpha, \beta) &= p(S|Z, \Phi)p(\Phi|\beta)p(Z|\Theta)p(\Theta|\alpha) \\ p(S|Z, \Phi) &= \prod_{d=1}^{M} p(s_d|z_d, \Phi) \\ p(Z|\Theta) &= \prod_{d=1}^{M} p(z_d|\Theta). \end{aligned} \tag{1}$$

Without loss of generality, we assume that s_d is a first-order Markov chain. Under this assumption, item $v_{d,j}$ has potential dependencies on the latest item $v_{d,j-1}$ and $v_{d,j}$ is generated by polynomial distribution $\phi_{z_d, v_{d,j-1}}$:

$$\begin{aligned} p(s_d|z_d, \Phi) &= \prod_{j=1}^{n_d} p(v_{d,j}|v_{d,j-1}, z_d, \Phi) \\ p(v_{d,j}|v_{d,j-1}, z_d, \Phi) &= \prod_{i=1}^{N} \phi_{z_d,i}^{\delta_{i,v_{d,j}}}, \end{aligned} \tag{2}$$

where n_d represents the length of session s_d. $\delta_{a,b}$ denotes the Kronecker delta function, where when $a = b$, the function returns 1, otherwise the function returns 0.

For each intention, we draw an intention-specific item distribution $\phi_{k,r} \sim \mathrm{Dir}(\beta)$. We denote $\beta^k_{(.),r} = \{\beta_{k,r.l}\}_{l=1}^N$ and $\eta^k_{(.),r} = \{n_{k,r.l}\}_{l=1}^N$. Then, we extract an intention distribution $\theta \sim \mathrm{Dir}(\alpha)$ for each session as follows:

$$\begin{aligned} p(\Phi|\beta) &= \prod_{k=1}^{K}\prod_{r=0}^{N} p(\phi_{k,r}|\beta_{k,r}) \\ &= \prod_{k=1}^{K}\prod_{r=0}^{N} \frac{\Gamma(\sum_{l=1}^{N}\beta_{k,r.l})}{\prod_{l=1}^{N}\Gamma(\beta_{k,r.l})} \prod_{l=1}^{N} \varphi_{k,r.l}^{\beta_{k,r.l}-1}, \end{aligned} \tag{3}$$

$$p(\Theta|\alpha) = \frac{\Gamma(\sum_{k=1}^{N}\alpha_k)}{\prod_{k=1}^{K}\Gamma(\alpha_k)} \prod_{k=1}^{K} \theta_k^{\alpha_k}, \tag{4}$$

$$p(S|Z, \Phi) = \prod_{d=1}^{M} p(s_d|z_d, \Phi), \tag{5}$$

$$p(S, Z|\alpha, \beta) = \frac{\Delta(\mathbf{T} + \mathbf{A})}{\Delta(\mathbf{A})} \left(\prod_{k=1}^{K}\prod_{r=0}^{N} \frac{\Delta(\eta^k_{(.),r} + \beta^k_{(.),r})}{\Delta(\beta^k_{(.),r})} \right). \tag{6}$$

Δ function is the normalization factor $\frac{\Gamma(\sum_{i=1}^{I} x_i)}{\prod_{i=1}^{I}\Gamma(x_i)}$. To infer $\phi_{k,r}$ and θ_k, we can adopt Gibbs sampling to perform approximate inference [6] and estimate the parameters as follows:

$$\begin{aligned} &p(z_d = k|Z_{-d}, S) \\ &\propto \frac{(\tau_k + \alpha - 1)}{M - 1 + K\alpha} \cdot \prod_{r=0}^{N} \left(\frac{\prod_{s=1}^{N}\prod_{i=1}^{N_d^{r,l}}(n_{k,r.l} + \beta_{k,r.l} + i - 1)}{\prod_{j=1}^{N_d^r}(n_{k,r} + N\beta_{k,r} + j - 1)} \right), \end{aligned} \tag{7}$$

where τ_k is the number of sessions owned by intention k. In addition, we have $\mathbf{T} = \{\tau_k\}_{k=1}^K$, $\mathbf{A} = \{\alpha_k\}_{k=1}^K$ and $\alpha_k = \alpha$. For jth token of the dth session, $v_{d,j-1} = r$, $v_{d,j} = l$. $r.l$ represents the item-pair and $n_{k,r.l}$ represents the times intention k associated with the pair $r.l$ in the whole data. n_r^k denotes the occurrences of pair r in intention k.

- For each session $s_d = \{v_{d,1}, v_{d,2} \ldots, v_{d,n_{d-1}}, v_{d,n_d}\}$, sample the intention-mixture components $\theta_d \sim \text{Dirichlet}(\alpha)$
- For each intention $k \in \{1, \ldots, K\}$ and item $r \in \{0, \ldots, N\}$
  – sample item selection components $\phi_{k,r} \sim \text{Dirichlet}(\beta_{k,r})$
- For each session $d \in 1, \ldots, M$ and $j \in 1, \ldots, N_d$
  – sample a intention $z_{d,j} \sim \text{Discrete}(\theta_d)$
  – sample a item $v_{d,j} \sim \text{Discrete}(\phi_{d,j}, v_{d,j-1})$.

Given Z, the parameters Θ and Φ can be estimated according to the following equations:

$$\theta_k = \frac{\tau_k + a_k}{\sum_{k'=1}^K (m_{k'} + \alpha_{k'})}, \phi_{k,r.l} = \frac{n_{k,r.l} + \beta_{k,r.l}}{\sum_{l'=1}^N (n_{k,r.l'} + \beta_{k,r.l'})}. \tag{8}$$

The computation of the data likelihood is critical in the inference and estimation. In general, the likelihood function is defined as:

$$\begin{aligned} p(S) &= \prod_{d=1}^M p(s_d) = \prod_{d=1}^M p(s_{d,1}, s_{d,2}, \ldots s_{d,n_d}) \\ &= \prod_{d=1}^M \sum_{k=1}^K p(s_{d,1}, s_{d,2}, \ldots s_{d,n_d}, z_d = k). \end{aligned} \tag{9}$$

Now, each model differs in the way the $p(v_{d,1}, .v_{d,N_d}, z_{d,N_d})$ component is defined. Bayes rule and the first-order Markov assumption over tokens simplifies the above probability into:

$$log p(S) = \sum_{d=1}^M log \left(\prod_{d=1}^M \sum_k \theta_k \phi_{k, v_{d,j-1}.v_{d,j}} \right). \tag{10}$$

Figure 3 shows the graphical intention model.

4.2 Neighbor Session Identification based on Intention Similarity

After obtaining the probability distribution $\mathbf{P}_{s_d} \in \mathbb{R}^K$ of the intentions of a session, we can calculate the similarity between the sessions using Cosine similarity according to Equation (11).

$$In_{sim} = \frac{\mathbf{P}_{s_o} \cdot \mathbf{P}_{s_d}}{||\mathbf{P}_{s_o}||||\mathbf{P}_{s_d}||} \tag{11}$$

For the target session s_o, we find its most similar previous sessions to constitute the neighbor set N_{s_o} based on the intention model. These sessions provide richer information.

5 EMBEDDING AND RELEVANCE SCORE PREDICTION IN SMONE

5.1 Item Embedding Based on GNN

A graph is suitable for representing the dependencies and interrelationships between various entities. Therefore, it is suitable for modeling session information. We model the items in the target session s_o as a session graph $\mathcal{G}_{s_o} = \{V_{s_o}, E_{s_o}\}$ where V_{s_o} is the node set and E_{s_o} is the edge set, in which each edge $e_{v_i,v_j} \in E_{s_o}$ denotes the pairwise transitions between items in a session. A graph is suitable for representing the dependencies and interrelationships between various entities. Therefore, it is suitable for modeling session information. We model the items in the

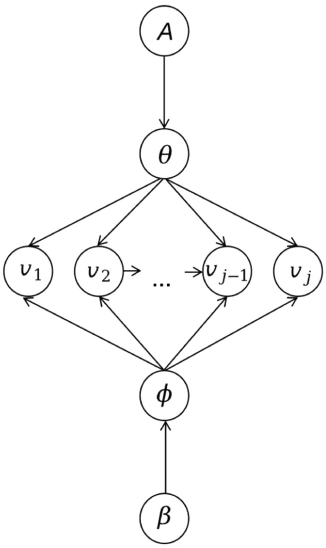


Fig. 3. Graphical intention models.

target session s and items of its neighbor sessions in N_s as an extended graph $EG_s = \{V_s, E_s\}$ where V_s is the node set and E_s is the edge set. Each edge $e_{v_i,v_j} \in E_s$ denotes the neighbor relationships between items in a session.

The session graph can encode the item sessions in a target session. The convolution operations for the ith item in the session s_o are as follows:

$$\mathbf{a}^t_{s_o,i} = \mathbf{A}_{s_o,i}\left[\mathbf{v}^{t-1}_{s_o,1}, \ldots, \mathbf{v}^{t-1}_{s_o,n}\right]^T \mathbf{H} + \mathbf{b}_g, \tag{12}$$

$$\mathbf{z}^t_{s_o,i} = \sigma\left(\mathbf{W}_z \mathbf{a}^t_{s_o,i} + \mathbf{U}_z \mathbf{v}^{t-1}_{s_o,i}\right), \tag{13}$$

$$\mathbf{r}^t_{s_o,i} = \sigma\left(\mathbf{W}_r \mathbf{a}^t_{s_o,i} + \mathbf{U}_r \mathbf{v}^{t-1}_{s_o,i}\right), \tag{14}$$

$$\widetilde{\mathbf{v}^t_{s_o,i}} = tanh\left(\mathbf{W}_e \mathbf{a}^t_{s_o,i} + \mathbf{U}_e(\mathbf{r}^t_{s,i} \odot \mathbf{v}^{t-1}_{s_o,i})\right), \tag{15}$$

$$\mathbf{v}^t_{s_o,i} = \left(1 - \mathbf{z}^t_{s_o,i}\right) \odot \mathbf{v}^{t-1}_{s_o,i} + \mathbf{z}^t_{s_o,i} \odot \widetilde{\mathbf{v}^t_{s_o,i}}, \tag{16}$$

where $\mathbf{H}$ is the weight. $\mathbf{v}^t_{so_j}(j = 1, 2, \ldots, n)$ stands for item embeddings with dimension d in tth step. $\mathbf{z}^t_{s_o,i}$ and $\mathbf{r}^t_{s_o,i}$ are the reset and update gates, respectively. $\mathbf{a}^t_{s_o,i}$ contains activations from edges in both directions. $\mathbf{b}_g$ is a bias parameter vector. $\sigma(\cdot)$ represents the sigmoid function and $\odot$ represents the element-wise multiplication function. $\mathbf{A}_{s_o} \in \mathbb{R}^{n\times 2n}$ contains $\mathbf{A}^{in}_{s_o}$ and $\mathbf{A}^{out}_{s_o}$ in two columns of blocks. $\mathbf{A}^{in}_{s_o} \in \mathbb{R}^{n\times n}$ is the adjacency matrix of nodes, which represents the weighted coefficients of incoming edges between nodes. $\mathbf{A}^{out}_{s_o}$ is the outgoing edge matrix. $\mathbf{A}_{s_o,i} \in \mathbb{R}^{1\times 2n}$ is the adjacency matrix corresponding to node i in the session.

We can obtain the current item embeddings using this method. The nodes' features in the current session are aggregated via the edges in the graph. The network outputs more typical representations of these nodes concluding the information of the current session.

ALGORITHM 1: Algorithm for Intention Modeling for Sessions

Input: The session set S;
Output: Intention assignment of each session $\mathbf{P}_{s_d}$;

Initialize $\tau_k, n^k_{(.),r.l}, n^k_{(.),r}$ to zeros;
for each session s_d in S **do**
  Sample an intention k:
  τ_k+= 1
  $z_d = k \sim Multi(1/K)$
  N_d = length of session s_d
  for $i \in [1, N_d]$ **do**
    $n^k_{(.),r.l}$ += 1
    $n^k_{(.),r}$ += 1
  end for
end for
repeat
  for session s_d **do**
    $k = z_d$//Record the intention of s_d
    $\tau_k - = 1\ z_d = k' \sim Multi(1/K)$
    for $i \in [1, N_d]$ **do**
      $n^k_{(.),r.l} - = 1$
      $n^k_{(.),r} - = 1$
    end for
    Sample an intention for $s_d\ z_d = k \sim p(z_d = k | Z_{-s_d}, S)$
    $\tau_k + = 1$
    for $i \in [1, N_d]$ **do**
      $n^k_{(.),r.l} + = 1$
      $n^k_{(.),r} + = 1$
    end for
  end for
until convergence
for $k \in [1, K]$ **do**
  $p(z_d = k) \sim p(z_d = k | Z_{-s_d}, S)$
end for
Output $\mathbf{P}_{s_d}$ for every session in S

5.2 Contextualized Item Embedding Based on a Hypergraph

A hypergraph can encode high-order data correlations (beyond pairwise connections) using its degree-free hyperedges [11]. The items in the neighbor sessions can be represented by a hypergraph and item-transitions would proceed in terms of intention. At the same time, these items acquire transitions not only from those items having a causal relationship with them but also from the items in cross-session.

We model the items in current session s_o and items of its neighbor sessions in N_{s_o} as a session hypergraph $\mathcal{H}_{s_o} = \{V_{s_o}, E_{s_o}\}$, where V_s is the node set and E_{s_o} is the edge set, in which each hyperedge $e_{s_o,i} \in E_{s_o}$ denotes a session and V_{s_o} represents the set of items in the target session and neighbor sessions. An example of a session hypergraph is shown in Figure 4.

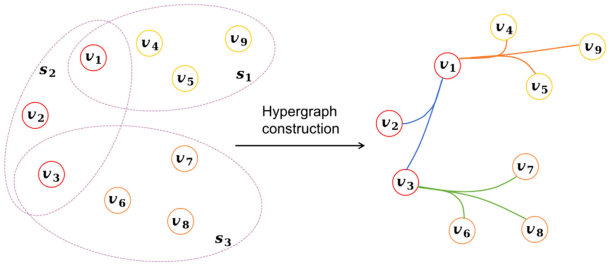


Fig. 4. An example of session hypergraph.

The hypergraph convolutional layer can encode the hypergraph-based cross-session item embedding. The convolution operation is:

$$\mathbf{s}_h^{'(nl+1)} = \mathbf{D}_h^{-1}\mathbf{H}_h\mathbf{W}_h\mathbf{B}_h^{-1}\mathbf{H}_h^T\mathbf{s}_h^{'(nl)}\mathbf{P}_h, \tag{17}$$

where $\mathbf{P}_h \in \mathbb{R}^{(nl)\times(nl+1)}$ is the weight matrix between layer nl and layer $nl+1$. $\mathbf{D}_h$ is the degree matrices of the vertex, and $\mathbf{B}_h$ is the degree matrices of the hyperedge, $\mathbf{H}_h \in \mathbb{R}^{nN\times nM}$ is the incidence matrix, and $\mathbf{W}_h \in \mathbb{R}^{nM\times nM}$ is the positive weight matrix. The weight value of each hyperedge is set to 1 equally. $\mathbf{s}_h^{'} = \{\mathbf{v}_{s_o,1}^h, \ldots, \mathbf{v}_{s_o,n}^h\}$ is the item embeddings in the sessions, and $\mathbf{v}_{s_o}^h$ denotes the item embeddings in the hypergraph channel. We can obtain the contextualized item embeddings using the hypergraph module.

5.3 Item Embedding Fusion

$\mathbf{v}_{s_o,i}^{'}$ and $\mathbf{v}_{s_o,i}^h$ denote item embeddings from the current session and the context channel, respectively. $\mathbf{v}_{s_o,i}^c$ represents the contextualized item embeddings in the features. Since both embeddings carry fundamental information, we fuse the embeddings of the nodes as Equation (18):

$$\mathbf{v}_{s_o,i}^c = tanh\left(\mathbf{W}_f\left(\mathbf{v}_{s_o,i}^{'} \| \mathbf{v}_{s_o,i}^h\right)\right). \tag{18}$$

$\|$ denotes the concatenation operation. $\mathbf{W}_f \in \mathbb{R}^{(nl)\times(nl+1)}$ is a parameter matrix, which represents a linear layer.

5.4 Session Embedding

We adopt the same strategy that is adopted in gated GNN [43] to represent the embedding of a session: $\mathbf{s}' = \{\mathbf{v}'_{s_o,1}, \mathbf{v}'_{s_o,2}, \ldots, \mathbf{v}'_{s_o,n}\}$. At the same time, $\mathbf{s}'_c = \{\mathbf{v}_{s_o,1}^c, \mathbf{v}_{s_o,2}^c, \ldots, \mathbf{v}_{s_o,n}^c\}$ denotes context-fused embeddings. The total session embedding is calculated as follows:

$$\mathbf{a}_i = \mathbf{f}^T\sigma\left(\mathbf{W}_1\upsilon_{s_o,n}^{'} + \mathbf{W}_2\upsilon_{s_o,i}^{'} + \mathbf{W}_3\upsilon_{s_o,i}^c + \mathbf{b}_i\right), \tag{19}$$

$$\mathbf{s}_g = \sum_{t=1}^{n}\mathbf{a}_i\mathbf{v}_{s_o,i}^{'}, \tag{20}$$

$$\mathbf{s}_h = \mathbf{W}_4\left[\mathbf{v}_{s_o,n}^{'} \| \mathbf{s}_g\right], \tag{21}$$

where $\mathbf{s}_g$ denotes the embedding of the session. $\mathbf{v}_{s_o,n}^{'}$ is the last item embedding in the current session. $\mathbf{v}_{s_o,i}^{'}$ is the item embedding in the current session. $\mathbf{v}_{s_o,i}^c$ is the item embedding in the session after fusing the neighbor information. Introducing parameter $\mathbf{f} \in \mathbb{R}^d$ and $\mathbf{W}_1, \mathbf{W}_2$, and $\mathbf{W}_3 \in \mathbb{R}^{d\times d}$ in Equation (19) means these item embeddings have different levels of priorities. $\mathbf{s}_h$ is the primary session embeddings except the last-clicked item. $\mathbf{b}_i$ is bias for item i. $\mathbf{a}_i$ is the weight. The final embedding of the session is $\mathbf{s}_h$, which can be calculated using Equation (21).

Table 2. Statistics of Datasets Used in the Experiments

Statistics	Yoochoose1/64	Diginetica	Tmall
click	557,248	982,961	818,479
train	369,859	719470	351,268
test	55,898	60,858	89,824
items	16,766	43,097	40,728
Average length	6.16	5.12	6.69

5.5 Item Recommendation

After obtaining the embedding of the target session, we multiply each candidate item vector $\mathbf{v}_i$ by $\mathbf{s}_h$ to compute the predicted scores of the items, which can be calculated as:

$$\hat{\mathbf{z}_i} = \mathbf{v}_i^T \mathbf{s}_h. \tag{22}$$

Then, we use a softmax function to generate the output vector $\hat{\mathbf{Y}}$:

$$\hat{\mathbf{Y}} = softmax(\hat{\mathbf{Z}}), \tag{23}$$

$$\mathbb{L}(\hat{\mathbf{y}}) = -\sum_{i=1}^{N} \mathbf{y}_i log(\hat{\mathbf{y}_i}) + (1 - \mathbf{y}_i) log(1 - \hat{\mathbf{y}_i}), \tag{24}$$

where $\mathbf{y}$ denotes the one-hot encoding vector of the ground truth item. Finally, we adopt the **back propagation through time (BPTT)** algorithm to train the model.

6 EXPERIMENT

We evaluate SMONE and compare it with other models on three real-world datasets and answer the following questions:

- **RQ1** How does SMONE perform compared with the representative recommendation approaches?
- **RQ2** How does the number of intentions impact the performance of SMONE?
- **RQ3** How does the number of neighbor sessions impact the performance of SMONE?
- **RQ4** How do the dimensions of embeddings impact the performance of SMONE?
- **RQ5** How does the depth of the model impact the performance of SMONE?
- **RQ6** Do the neighbor sessions benefit improving the recommendation performance?
- **RQ7** How is the model training time affected by certain factors?

6.1 Datasets

Our experiments are performed on three real-world datasets, namely, Yoochoose [15], Diginetica [21], and Tmall [46]. Yoochoose was built for the RecSys 2015 competition, and it contains a collection of sessions, which represents the users' clicks on an e-commerce website. The Diginetica dataset was prepared for the CIKM CUP 2016 and consists of transactional data only. The Tmall dataset was used for the IJCAI-15 competition. It contains anonymous shopping logs on the Tmall online shopping platform. The statistics of the datasets are listed in Table 2.

6.2 Experimental Settings

In the experiment, we use PyTorch libraries and Torch.geometric libraries to implement our model. All the experiments were performed using TITAN RTX with 126 GB RAM.

We set the dimensionality to 100 of the embedding vectors for all datasets. The parameters in the model are initialized with a mean of 0 and a standard deviation of 0.1. For the Adam optimizer,

the initial learning rate is set to 0.001 with a decay of 0.1 every 3 epochs. The batch size was set to 128 and the L2 penalty was set to 10^{-5}.

6.3 Metrics

We use two metrics to evaluate SMONE and other comparative algorithms. The two metrics are as follows:

P@N is the prediction accuracy, which represents the proportion of correctly recommended items among the top-N items.

MRR@N is the mean reciprocal rank. It is the average reciprocal of the position of the recommend item when the recommendation is correct. The higher the MRR@N value, the better the prediction.

Specifically, we use P@10, P@20, MRR@10, and MRR@20 as the evaluation metrics in this article. For significance testing, we use a paired t-test with $p < 0.05$.

6.4 Comparative Methods

We compare SMONE with the following representative methods to evaluate its performance:

FPMC [49]: FPMC combines a first-order Markov chain model and matrix factorization for next basket recommendation.

GRU4REC [18] and **NARM** [21]: These are both RNN- based methods. GRU4REC uses a session-parallel mini-batch training process to model user click sequence. The number of hidden units was set to 100. The dropout parameter was set to 0. The learning rate was 0.01. The mini-batch was fixed at 500. NARM employs the attention mechanism to capture the user's main purpose and combines it with the sequence behavior as the final representation for the session-based recommendation. The number of hidden units was set to 100. The dropout parameter was set to 0.25. The learning rate was 0.001. The mini-batch was fixed at 512. The number of epochs was set to 30.

STAMP [24]: It is an attention-based method. STAMP is a novel short-term memory priority model that captures user's general preference and current interests according to employing simple MLP networks and an attentive net. The learning rate was 0.005 and the learning rate decay was 1.0. The mini-batch was fixed at 512. The number of epochs was set to 30.

CRSM [37]: CRSM applies collaborative neighborhood information to session-based recommendations. We set two momentum parameters $\beta 1 = 0.9$ and $\beta 2 = 0.999$, respectively. The learning rate was 0.0001 and the batch size was set as 512. The embedding dimension of items and hidden units of the GRU were 100. The number of nearest neighbors was 256.

SR-GNN and **DGTN** [53]: These are both two GNN-based methods. SR-GNN was the first to use a graph network to establish relationships between items within a session. The learning rate was set to 0.001 and the decay rate was 0.1 after every 3 epochs. The batch size and the L2 penalty was set to 100 and 10^{-5}, respectively. The number of epochs was set to 30. Different from SR-GNN, DGTN calculates the repetition of items within a session to find the neighbor sessions and then introduces cross-session information to improve the performance of the model. The learning rate was set to 0.001 and the decay rate was 0.1 after every 3 epochs. The batch size was set to 100. The number of epochs was set to 30 and the number of neighbor sessions was 140.

FGNN: FGNN collaboratively considers the sequence order and the latent order in the session graph for a session-based recommender system [30]. It formulates the next item recommendation within the session as a graph classification problem. The learning rate was 0.001 and the learning rate decay was 0.1 after every 3 epochs. The batch size and the L2 penalty were fixed at 100 and 10^{-5}. The number of epochs was set to 30.

ICM-SR [20]: ICM-SR is similar to our method on leveraging prior items in neighbor sessions. It encodes current session by leveraging the previous items and the last item to represent the session,

Table 3. Experiment Results of Different Methods

Method	YOOCHOOSE1/64				DIGINETICA				Tmall			
	P@10	MRR@10	P@20	MRR@20	P@10	MRR@10	P@20	MRR@20	P@10	MRR@10	P@20	MRR@20
FMPC	37.41	20.01	45.64	15.89	15.48	6.24	23.53	6.73	15.48	6.24	16.08	7.32
GRU4REC	52.39	24.51	60.11	22.57	17.87	7.66	29.45	8.33	17.87	7.66	10.87	5.85
NARM	57.77	27.34	68.71	28.65	35.67	15.24	49.70	16.13	19.22	10.71	23.43	11.01
STAMP	58.89	29.12	68.72	29.60	33.94	14.26	45.61	14.24	22.66	13.18	26.49	13.38
FGNN	59.13	29.66	70.03	30.33	37.72	15.95	50.36	16.68	23.57	12.62	26.24	12.88
SR-GNN	59.98	29.94	70.51	30.48	36.86	15.52	50.70	17.51	24.39	13.41	27.57	13.72
CRSM	59.92	30.02	70.79	30.48	35.52	15.52	50.00	17.16	24.61	13.53	25.11	13.68
DGTN	58.84	30.11	70.83	30.33	38.33	15.99	51.38	17.78	23.87	13.64	28.01	13.97
ICM-SR	60.27	30.63	70.08	31.17	38.89	16.21	52.33	17.72	24.84	13.64	28.59	14.01
SMONE-BTM	60.36	30.71	71.13	31.28	39.08	16.58	52.41	17.95	24.57	13.78	28.61	13.97
SMONE-MOU	60.39	30.75	71.15	31.29	39.11	16.62	52.46	18.09	24.76	13.81	28.63	14.15
SMONE-No	59.88	29.89	70.64	30.51	36.89	15.55	50.75	17.54	23.43.	13.47	27.60	13.74
SMONE	**60.43**	**30.83**	**71.23**$^{\triangle}$	**31.37**$^{\triangle}$	**39.15**	**16.74**	**52.52**$^{\triangle}$	**18.13**$^{\triangle}$	**24.96**	**13.83**	**28.69**$^{\triangle}$	**14.16**$^{\triangle}$

$\triangle$denotes a significant improvement of SR-IEM over the other baseline using a paired t-test ($p < 0.05$)

.

Table 4. Experiment Results with Different Intention Numbers

Intention number	YOOCHOOSE1/64				DIGINETICA				Tmall			
	P@10	MRR@10	P@20	MRR@20	P@10	MRR@10	P@20	MRR@20	P@10	MRR@10	P@20	MRR@20
\10	59.93	29.76	70.03	30.65	37.83	16.17	51.22	17.86	24.51	13.56	28.06	13.83
\20	60.11	29.96	70.32	30.82	38.01	16.35	51.61	17.98	24.53	13.62	28.17	14.05
\30	60.23	30.33	70.85	31.04	38.19	16.50	51.70	18.07	24.63	13.67	28.35	14.09
\40	60.35	30.71	71.06	31.19	38.46	16.61	52.25	18.06	24.71	13.75	28.53	14.11
\50	**60.43**	**30.83**	**71.23**	**31.37**	38.62	16.73	**52.52**	**18.13**	24.78	**13.83**	**28.69**	**14.16**
\60	60.38	30.77	71.14	31.28	**39.15**	**16.74**	52.46	18.08	**24.96**	13.79	28.63	14.12

which is then used to produce initial item predictions as intent. The session with similar intent will be defined as the neighbor session. The learning rate was 0.001 and the learning rate decay was 0.1 after every 3 epochs. The batch size and the L2 penalty are fixed at 100 and 10^{-5}. The number of epochs was set to 30. The number of candidate neighbors and the final neighbors are set to 1,000 and 100, respectively.

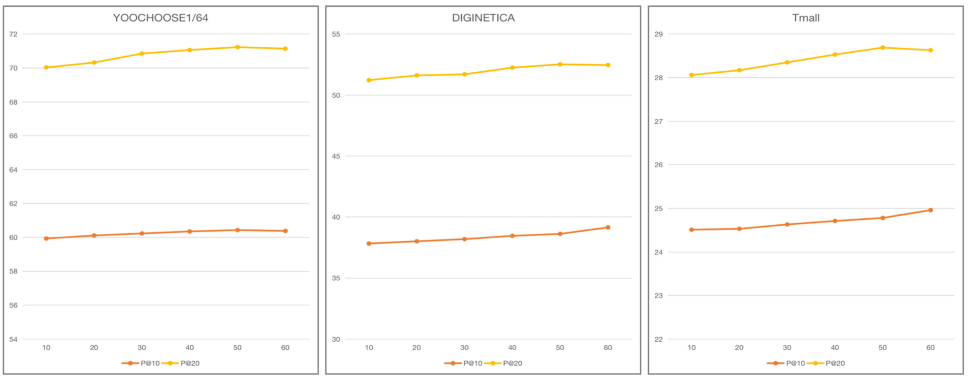
6.5 Experiment Results and Discussion

6.5.1 Overall Performance (RQ1). The experimental results of all the methods are shown in Tables 3, 4, 5. It can be noted both the RNN-based methods (GRU4REC and NARM) underperform their neighborhood-based variants (KNN-RNN). This shows that adopting collaborative information from other sessions is an effective method.

- Deep learning methods show their strength in session-based recommendation. Their performance is clearly superior to traditional recommendation methods, which proves that adopting a deep learning technology in recommendation systems is a necessary means, since neural networks can model the complex interactions among items.
- In general, the graph-based methods (SR-GNN, DGTN, FGNN, and SMONE) outperform the RNN-based methods (GRU4REC, NARM, KNN-RNN, and CSRM). This also verifies the conclusion that modeling the relationship of items as graph-structured models by using GNN is superior between items, as graph-structured models using GNN are superior to the RNN-based method.
- We compared our SMONE with other variants by use of topic models to model intentions directly. SMON-BTM and SMONE-MOF are the SMONE variants utilizing biterm topic model and mixture of unigrams to extract intentions of sessions, respectively. It can be found SMONE achieves the best performance on all of the datasets. By learning the context embedding, SMONE has the ability to fuse information of items within corresponding sessions more efficiently.

Table 5. Experiment Results with Different Neighbor Session Numbers

Neighbor Session Number	Yoochoose 1/64			
	P@10	MRR@10	P@20	MRR@20
5	59.90	29.92	70.64	30.53
10	59.93	29.93	70.66	30.55
15	59.95	29.98	70.69	30.61
20	60.01	30.06	70.71	30.68
25	60.08	30.13	70.79	30.85
30	60.16	30.28	70.89	30.96
35	60.22	30.43	70.98	31.05
40	60.29	30.52	71.12	31.17
45	60.32	30.68	71.19	31.27
50	60.43	30.83	71.23	31.37
55	60.36	30.72	71.15	31.28
60	60.30	30.59	71.09	31.19

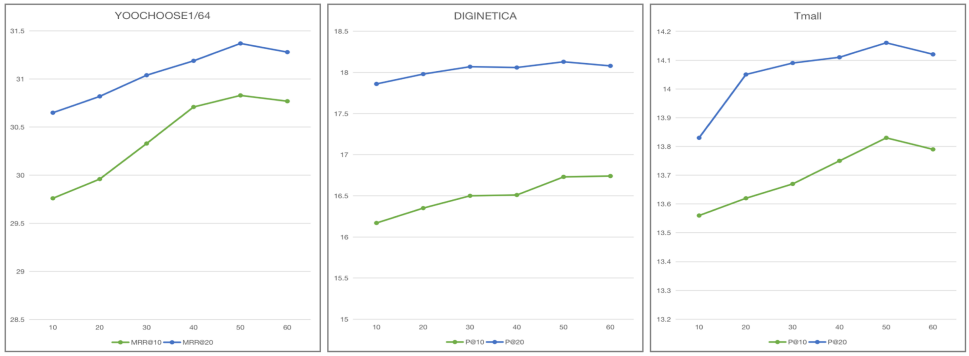


(a) P@K on YOOCHOOSE1/64 (b) P@K on DIGINETICA (c) P@K on TMALL

Fig. 5. P@K with different intention numbers.

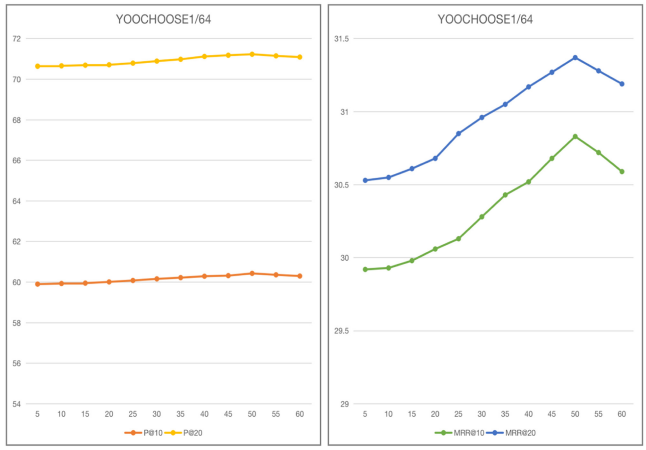
6.5.2 Impact of the Number of Intentions (RQ2). To analyze the impact of the number of intentions, we change the number of intentions from 10 to 60. Table 4 and Figures 5 and 6 show the results. The k value of top-k neighbors is set as 50. On the Yoochoose1/64 dataset, P@10, P@20, MRR@10, and MRR@20 improve with the number of intentions until the number of intentions is 50. On the DIGINETICA dataset, P@20 and MRR@20 also increase with the number of intentions, and the best result is when the number of intentions is 50. But P@10 and MRR@10 of the model performance achieve the best result until the number of intentions is 60. We also test the model performance on Tmall dataset. And we obtain similar result on the Yoochoose1/64 except for P@10. We did not test the performance of our model with the number of intentions larger than 60, because generating intentions by the intention model takes more than 5 hours when the number of topics is 60. This indicates that the correct clustering of neighbor sessions benefits improves the performance of SMONE.

6.5.3 Impact of the Number of Neighbor Sessions (RQ3). In the experiment above, the k value of the top-k neighbors is set as 50. But the number of neighbor sessions indirectly determines how much additional semantic information from the global can be provided with. For a large k,



(a) MRR@K on YOOCHOOSE1/64 (b) MRR@K on DIGINETICA (c) MRR@K on TMALL

Fig. 6. MRR@K with different intention numbers.



(a) P@K (b) MRR@K

Fig. 7. Experiments results with different neighbor sessions on Yoochoose1/64.

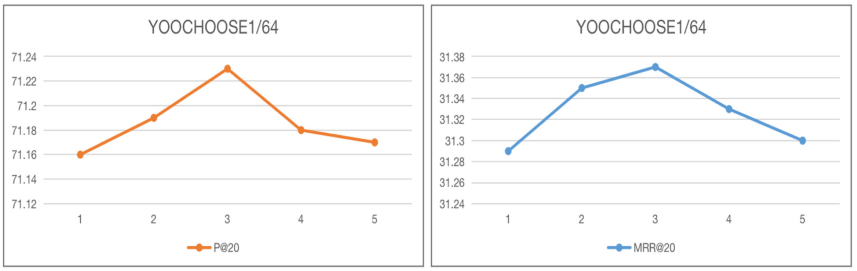
to construct the hypergraph and learn the embeddings takes a great amount of time. Therefore, to test the impact of numbers of neighbor sessions, we select the Yoochoose1/64 dataset and the numbers of intention in the experiments here are all set to 50.

The k values are selected from 5 to 60 to test the impact of the number of neighbor sessions. The results are listed in Table 5 and Figure 7. It can be seen that neither too many neighbor sessions nor too few perform optimally. When the number of neighbor sessions is 50, the performance of the model is better than the other model with a different number of neighbor sessions. Model with 50 neighbor sessions achieves the best performance measured with all the metrics. It is very important to select the proper number of neighbor sessions for the model. A too-small number of neighbor sessions cannot provide enough additional information, while too many sessions tend to generate noise.

6.5.4 Impact of Dimensions of Embeddings (RQ4). We compare the performance of the model with different embeddings of dimension. We change the number of dimensions from 50 to 200.

Table 6. Experiment Results with Different Dimensions of Embeddings

	Yoochoose1/64							
	50		100		150		200	
	P@20	MRR@20	P@20	MRR@20	P@20	MRR@20	P@20	MRR@20
\SMONE	31.13	71.14	31.37	71.23	31.12	71.08	31.10	71.04



(a) P@20 with Different Layers (b) MRR@20 with Different Layers

Fig. 8. Impact of depth.

The model achieves the best results when the number of dimensions of the embeddings is 100. The results are shown in Table 6.

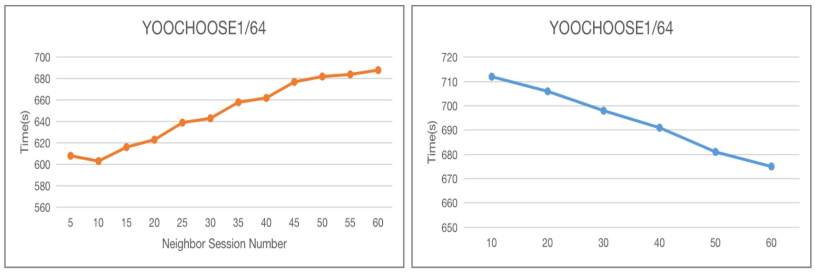
6.5.5 Impact of Depth (RQ5). We also consider the impact of depth of the model. The numbers of layers of the network range between 1, 2, 3, 4, 5. The model achieves the best result with 3 hypergraph convolution layers. Too many or too few layers can reduce the performance of the model. The results are as shown in Figure 8.

6.5.6 Ablation Study (RQ6). In addition, we conducted an ablation experiment on the variant model without the context channel. The variant SMONE-No is similar to SR-GNN except for a jumping knowledge layer. The input of the target session channel and the output of this channel aggregate with the jumping knowledge strategy. The representations of nodes from different layers comprise the representation v'_s through max pooling and the P@10, MRR@10, P@20, and MRR@20 of the variant on Yoochoose1/64 are 59.88, 29.89, 70.64, and 30.51, respectively. The same phenomenon was observed on the other two datasets. SMONE achieves the best performance on all of the datasets in the experiment compared with SMONE-No. By learning the context embedding, SMONE has the ability to fuse information of items within corresponding sessions.

6.5.7 Model Training Efficiency (RQ7). To measure the computational efficiency of the model, we record their different training time with different neighbor session numbers and different intention numbers. In the experiments, we set the batch size as 128 and the hidden size as 100. The model is trained with 15 epochs, and we record the average training time per epoch.

In the experiment for training time testing with different neighbor session numbers, we set the intention number to 50. Figure 9(a) shows that training time increases with neighbor session number.

In the experiment for training time testing with different intention numbers, Figure 9(b) shows that the training time decreases with intention numbers. To understand the reason, we count the average numbers of neighbor sessions under different intention numbers. When the intention number increases from 10 to 60, the average number of neighbor sessions decreases. We believe that it is the reason for the gradual decrease of training time with the intention numbers.



(a) Training Time with Different Neighbor Session Numbers

(b) Training Time with Different Intention Numbers

Fig. 9. Computational efficiency.

7 CONCLUSIONS AND FUTURE WORK

We propose a novel method, SMONE, to explore more informative historical sessions to improve the performance of SRSs. To overcome the sparsity problem to find neighbor sessions in terms of the same item between the target session and historical sessions, we identify the neighbor sessions that have the similar intentions to the target session. After the neighbor sessions are found, the current session is constructed as a graph to learn the item embeddings. In addition, the target session and its neighbor sessions are modeled as a hypergraph to learn the contextualized item embeddings. Specifically, we use hypergraph convolution to embed the context information based on the extended session graph. Extensive experiments on three real-world datasets show SMONE achieves better performance than its counterparts. It also indicates intention-similar sessions do provide useful information to help improve the performance of SRSs.

Our work can be extended further. In this article, we recommend the next item mainly based on the current session and neighbor sessions to find information that can be used. How to define and search the neighbor sessions from all the sessions is a kind of future work. Moreover, how to model item correlations within different sessions is another challenge for future study.

REFERENCES

[1] Ngo Xuan Bach, Dang Hoang Long, and Tu Minh Phuong. 2020. Recurrent convolutional networks for session-based recommendations. *Neurocomputing* 411 (2020), 247–258.

[2] Ting Bai, Jian-Yun Nie, Wayne Xin Zhao, Yutao Zhu, Pan Du, and Ji-Rong Wen. 2018. An attribute-aware neural attentive model for next basket recommendation. In *Proceedings of the 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. 1201–1204.

[3] Bhagyashree Vyankatrao Barde and Anant Madhavrao Bainwad. 2017. An overview of topic modeling methods and tools. In *Proceedings of the International Conference on Intelligent Computing and Control Systems (ICICCS)*. IEEE, 745–750.

[4] Marco Bertini, Andrea Ferracani, Riccardo Papucci, and Alberto Del Bimbo. 2020. Keeping up with the influencers: Improving user recommendation in Instagram using visual content. In *Proceedings of the 28th ACM Conference on User Modeling, Adaptation and Personalization*. 29–34.

[5] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet allocation. *J. Mach. Learn. Res.* 3 (2003), 993–1022.

[6] Chris K. Carter and Robert Kohn. 1994. On Gibbs sampling for state space models. *Biometrika* 81, 3 (1994), 541–553.

[7] Tianwen Chen and Raymond Chi-Wing Wong. 2020. Handling information loss of graph neural networks for session-based recommendation. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1172–1180.

[8] Tong Chen, Hongzhi Yin, Hongxu Chen, Rui Yan, Quoc Viet Hung Nguyen, and Xue Li. 2019. AIR: Attentional intention-aware recommender systems. In *Proceedings of the IEEE 35th International Conference on Data Engineering (ICDE)*. 304–315. DOI: https://doi.org/10.1109/ICDE.2019.00035

[9] Chuan Cui, Qi Shen, Shixuan Zhu, Yitong Pang, Yiming Zhang, Zhenwei Dong, and Zhihua Wei. 2021. Intention adaptive graph neural network for category-aware session-based recommendation. *arXiv preprint arXiv:2112.15352* (2021).

[10] Ramazan Esmeli, Mohamed Bader-El-Den, and Alaa Mohasseb. 2019. Context and short term user intention aware hybrid session based recommendation system. In *Proceedings of the IEEE International Symposium on INnovations in Intelligent SysTems and Applications (INISTA)*. IEEE, 1–6.

[11] Yifan Feng, Haoxuan You, Zizhao Zhang, Rongrong Ji, and Yue Gao. 2019. Hypergraph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 3558–3565.

[12] Lei Guo, Hongzhi Yin, Qinyong Wang, Tong Chen, Alexander Zhou, and Nguyen Quoc Viet Hung. 2019. Streaming session-based recommendation. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1569–1577.

[13] Priyanka Gupta, Diksha Garg, Pankaj Malhotra, Lovekesh Vig, and Gautam Shroff. 2019. NISER: Normalized item and session representations to handle popularity bias. *arXiv preprint arXiv:1909.04276* (2019).

[14] Balázs Hidasi and Alexandros Karatzoglou. 2018. Recurrent neural networks with top-k gains for session-based recommendations. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. 843–852.

[15] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2015. Session-based recommendations with recurrent neural networks. *arXiv preprint arXiv:1511.06939* (2015).

[16] Thomas Hofmann. 1999. Probabilistic latent semantic indexing. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. 50–57.

[17] Hyunwoo Hwangbo and Yangsok Kim. 2019. Session-based recommender system for sustainable digital marketing. *Sustainability* 11, 12 (2019), 3336.

[18] Dietmar Jannach and Malte Ludewig. 2017. When recurrent neural networks meet the neighborhood for session-based recommendation. In *Proceedings of the 11th ACM Conference on Recommender Systems*. 306–310.

[19] Wang-Cheng Kang and Julian McAuley. 2018. Self-attentive sequential recommendation. In *Proceedings of the IEEE International Conference on Data Mining (ICDM)*. IEEE, 197–206.

[20] Haoyang Li, Xin Wang, Ziwei Zhang, Jianxin Ma, Peng Cui, and Wenwu Zhu. 2021. Intention-aware sequential recommendation with structured intent transition. *IEEE Trans. Knowl. Data Eng.* (2021). DOI: https://doi.org/10.1109/TKDE.2021.3050571

[21] Jing Li, Pengjie Ren, Zhumin Chen, Zhaochun Ren, Tao Lian, and Jun Ma. 2017. Neural attentive session-based recommendation. In *Proceedings of the ACM on Conference on Information and Knowledge Management*. 1419–1428.

[22] Lin Liu, Li Wang, and Tao Lian. 2021. Discovering proper neighbors to improve session-based recommendation. In *Proceedings of the Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 353–369.

[23] Qiang Liu, Shu Wu, Diyi Wang, Zhaokang Li, and Liang Wang. 2016. Context-aware sequential recommendation. In *Proceedings of the IEEE 16th International Conference on Data Mining (ICDM)*. IEEE, 1053–1058.

[24] Qiao Liu, Yifu Zeng, Refuoe Mokhosi, and Haibin Zhang. 2018. STAMP: Short-term attention/memory priority model for session-based recommendation. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1831–1839.

[25] Muyang Ma, Pengjie Ren, Zhumin Chen, Zhaochun Ren, Lifan Zhao, Peiyu Liu, Jun Ma, and Maarten de Rijke. 2022. Mixed information flow for cross-domain sequential recommendations. *ACM Trans. Knowl. Discov. Data* 16, 4 (2022), 1–32.

[26] Kamal Nigam, Andrew Kachites McCallum, Sebastian Thrun, and Tom Mitchell. 2000. Text classification from labeled and unlabeled documents using EM. *Mach. Learn.* 39, 2 (2000), 103–134.

[27] Zhiqiang Pan, Fei Cai, Wanyu Chen, Chonghao Chen, and Honghui Chen. 2022. Collaborative graph learning for session-based recommendation. *ACM Trans. Inf. Syst.* 40, 4 (2022), 1–26.

[28] Zhiqiang Pan, Fei Cai, Yanxiang Ling, and Maarten de Rijke. 2020. An intent-guided collaborative machine for session-based recommendation. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1833–1836.

[29] Nan Qiu, BoYu Gao, Huawei Tu, Feiran Huang, Quanlong Guan, and Weiqi Luo. 2022. LDGC-SR: Integrating long-range dependencies and global context information for session-based recommendation. *Knowl.-based Syst.* 248 (2022), 108894.

[30] Ruihong Qiu, Jingjing Li, Zi Huang, and Hongzhi Yin. 2019. Rethinking the item order in session-based recommendation with graph neural networks. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. 579–588.

[31] Massimo Quadrana, Alexandros Karatzoglou, Balázs Hidasi, and Paolo Cremonesi. 2017. Personalizing session-based recommendations with hierarchical recurrent neural networks. In *Proceedings of the 11th ACM Conference on Recommender Systems*. 130–137.

[32] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2010. Factorizing personalized Markov chains for next-basket recommendation. In *Proceedings of the 19th International Conference on World Wide Web*. 811–820.

[33] Weiqi Shao, Xu Chen, Long Xia, Jiashu Zhao, and Dawei Yin. 2022. Sequential recommendation with user evolving preference decomposition. *arXiv preprint arXiv:2203.16942* (2022).

[34] Yanyan Shen, Baoyuan Ou, and Ranzhen Li. 2022. MBN: Towards multi-behavior sequence modeling for next basket recommendation. *ACM Trans. Knowl. Discov. Data* 16, 5 (2022), 1–23.

[35] Zhu Sun, Qing Guo, Jie Yang, Hui Fang, Guibing Guo, Jie Zhang, and Robin Burke. 2019. Research commentary on recommendations with side information: A survey and research directions. *Electron. Commerce Res. Applic.* 37 (2019), 100879.

[36] Fuyun Wang, Xuequan Lu, and Lei Lyu. 2022. CGSNet: Contrastive graph self-attention network for session-based recommendation. *Knowl.-based Syst.* 251 (2022), 109282.

[37] Meirui Wang, Pengjie Ren, Lei Mei, Zhumin Chen, Jun Ma, and Maarten de Rijke. 2019. A collaborative session-based recommendation approach with parallel memory modules. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 345–354.

[38] Nan Wang, Shoujin Wang, Yan Wang, Quan Z. Sheng, and Mehmet A. Orgun. 2022. Exploiting intra-and inter-session dependencies for session-based recommendations. *World Wide Web* 25, 1 (2022), 425–443.

[39] Shoujin Wang, Longbing Cao, Yan Wang, Quan Z. Sheng, Mehmet A. Orgun, and Defu Lian. 2021. A survey on session-based recommender systems. *ACM Comput. Surv.* 54, 7 (2021), 1–38.

[40] Shoujin Wang, Liang Hu, and Longbing Cao. 2017. Perceiving the next choice with comprehensive transaction embeddings for online recommendation. In *Proceedings of the Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 285–302.

[41] Zhitao Wang, Chengyao Chen, Ke Zhang, Yu Lei, and Wenjie Li. 2018. Variational recurrent model for session-based recommendation. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. 1839–1842.

[42] Ziyang Wang, Wei Wei, Gao Cong, Xiao-Li Li, Xian-Ling Mao, and Minghui Qiu. 2020. Global context enhanced graph neural networks for session-based recommendation. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 169–178.

[43] Shu Wu, Yuyuan Tang, Yanqiao Zhu, Liang Wang, Xing Xie, and Tieniu Tan. 2019. Session-based recommendation with graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 346–353.

[44] Shiwen Wu, Yuanxing Zhang, Chengliang Gao, Kaigui Bian, and Bin Cui. 2020. Garg: Anonymous recommendation of point-of-interest in mobile networks by graph convolution network. *Data Sci. Eng.* 5, 4 (2020), 433–447.

[45] Xiang Wu, Qi Liu, Enhong Chen, Liang He, Jingsong Lv, Can Cao, and Guoping Hu. 2013. Personalized next-song recommendation in online karaokes. In *Proceedings of the 7th ACM Conference on Recommender Systems*. 137–140.

[46] Xin Xia, Hongzhi Yin, Junliang Yu, Qinyong Wang, Lizhen Cui, and Xiangliang Zhang. 2020. Self-supervised hypergraph convolutional networks for session-based recommendation. *arXiv preprint arXiv:2012.06852* (2020).

[47] Chengfeng Xu, Pengpeng Zhao, Yanchi Liu, Victor S. Sheng, Jiajie Xu, Fuzhen Zhuang, Junhua Fang, and Xiaofang Zhou. 2019. Graph contextualized self-attention network for session-based recommendation. In *Proceedings of the International Joint Conference on Artificial Intelligence*, Vol. 19. 3940–3946.

[48] Xiaohui Yan, Jiafeng Guo, Yanyan Lan, and Xueqi Cheng. 2013. A biterm topic model for short texts. In *Proceedings of the 22nd International Conference on World Wide Web*. 1445–1456.

[49] Feng Yu, Qiang Liu, Shu Wu, Liang Wang, and Tieniu Tan. 2016. A dynamic recurrent model for next basket recommendation. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 729–732.

[50] Feng Yu, Yanqiao Zhu, Qiang Liu, Shu Wu, Liang Wang, and Tieniu Tan. 2020. TAGNN: Target attentive graph neural networks for session-based recommendation. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1921–1924.

[51] Chunkai Zhang, Quan Liu, and Zeyu Zhang. 2022. DSGNN: A dynamic and static intentions integrated graph neural network for session-based recommendation. *Neurocomputing* 468 (2022), 222–232.

[52] Wayne Xin Zhao, Jing Jiang, Jianshu Weng, Jing He, Ee-Peng Lim, Hongfei Yan, and Xiaoming Li. 2011. Comparing Twitter and traditional media using topic models. In *Proceedings of the European Conference on Information Retrieval*. Springer, 338–349.

[53] Yujia Zheng, Siyi Liu, Zekun Li, and Shu Wu. 2020. DGTN: Dual-channel graph transition network for session-based recommendation. In *Proceedings of the International Conference on Data Mining Workshops (ICDMW)*. IEEE, 236–242.

[54] Nengjun Zhu, Jian Cao, Yanchi Liu, Yang Yang, Haochao Ying, and Hui Xiong. 2020. Sequential modeling of hierarchical user intention and preference for next-item recommendation. In *Proceedings of the 13th International Conference on Web Search and Data Mining*. 807–815.

[55] Nengjun Zhu, Jian Cao, Xinjiang Lu, and Hui Xiong. 2021. Learning a hierarchical intent model for next-item recommendation. *ACM Trans. Inf. Syst.* 40, 2 (2021), 1–28.

Received 30 June 2022; revised 15 February 2023; accepted 27 February 2023